APEX Workflows

LANL, NERSC, SNL

SAND2015-10342 O
LA-UR-15-29113

November 24, 2015

1 Introduction

The Department of Energy (DOE) compute facilities operate as resources for the National Nuclear Security Administration (NNSA) and Office of Science. In this document we describe a subset of scientific workflows which are run on current platforms. There is significant commonality to how Los Alamos National Laboratory (LANL), Sandia National Laboratory (SNL), Lawrence Livermore National Laboratory (LLNL) and the National Energy Research Scientific Computing Center (NERSC) operate and conduct scientific inquiry with supercomputers. In addition to presenting the commonality between the sites, we believe that opportunities exist to significantly optimize the operation of our facilities and more rapidly advance our scientific goals. In describing how our scientific workflows move from inception to realization we believe that we can also enable significant improvements in how computer architectures support real scientific workflows.

1.1 Computational Campaign Overview

In order to use the computing resources at a compute facility, teams composed of domain experts and computer scientists develop and submit proposals describing the team's scientific goals and the details of how that site's supercomputers will be used (including the computational time and storage needed to accomplish the proposed scientific goals). All sites, except NERSC, select a number of proposals to execute on the site machine during a given *computing campaign* based on the quality of the submitted proposal and the suitability of the described scientific goals. The selected teams are granted computing allocations ranging from a few days of full system time to almost a full month of computational time based on the amount of time the team has requested and the amount of time granted to other approved teams during that campaign. Note there is not an expectation that all proposals will scale to the degree required to leverage the entire machine. Nor is there an expectation by the proposal teams that the machine will execute the team's jobs by some deadline other than within the campaign. Nevertheless, some proportion of the submitted jobs will leverage

the full machine, and some teams will advise the operators of deadlines that require allowing a series of jobs to finish before some non-campaign deadline.

Included below are two *workflow diagrams* that demonstrate two classes of computational science workloads: large-scale scientific simulation and large-scale high-throughput computing. The large rectangle at the top of the figure shows the data sets generated throughout the workflow and the timescale during which the data is retained. In particular, we have differentiated three timescales of interest: temporary, campaign, and forever. Data generated on temporary timescales includes checkpoints and analysis data sets that are produced by the application, but that the domain scientist will eventually discard (usually at job completion, or when a later data processing step completes). The campaign timescale describes data that is generated and useful throughout the execution of the entire scientific workflow. Once the campaign is complete the scientist will discard this data. Finally, we consider the data retention period labeled forever. Forever timescale data will persist longer than the machine used to generate the data. This typically includes small numbers of checkpoint data sets; however, this is primarily the analysis data sets generated by the application and re-processed by the user/scientist.

The lower portion of the diagram shows the *phases* of the workflow. A workflow phase is executed as a series of parallel and/or serial jobs submitted to the batch scheduler and executed when sufficient resources exist to execute the submitted jobs. The scheduler may maintain several job scheduling queues to ensure that high-value jobs, such as jobs constructed to complete the interactive analysis phase of the workflow, execute immediately (or perhaps during business hours when scientists will be at their terminals). Most phases of the workflow require executing tens and possibly thousands of jobs, with each job continuing the progress of prior jobs.

The workflow phases also describe a dependency between the submission of jobs into the scheduling queues. However, the strict ordering requirements are particular to the specific workflow phases. In some cases, a phase cannot begin until all of the prior phases are complete. In other cases, a phase cannot begin until the first *set of jobs* from the prior phases are complete; but there is no requirement that the prior phases be complete, only that some number of jobs are complete. We describe these dependencies in detail in sections 1.2 and 1.3.

Finally, we include in the workflow phase an icon for the initiator of the jobs submitted during the phase. The stick person icon indicates that a human is "in the loop" and performing job submissions. An instrument icon indicates that an external instrument or automated process is performing the job submissions. We use this icon to indicate that the computational resources are coupled in near real-time to an on-going experiment or orchestration framework. For example, compute jobs are automatically submitted to the scheduling queue when data is generated by LBNL's Advanced Light Source (ALS) beamlines.

1.2 Simulation Science Workflow Overview

Large-scale multi-physics simulation of real world phenomena are one of the critical workflows for APEX supercomputers. Although the application software packages for simulating diverse fields, such as relativistic shock waves, combustion, and plasma flows, differ based on the underlying physics modeled, a great deal of commonality exists across scientific sim-

ulation workflows. In particular, the APEX team has analyzed a large proportion of the simulation science workflows popular on existing supercomputers and has constructed an overview of how a scientist leverages multi-physics simulation software, parallel tools, and analysis software to improve the understanding of the physical world.

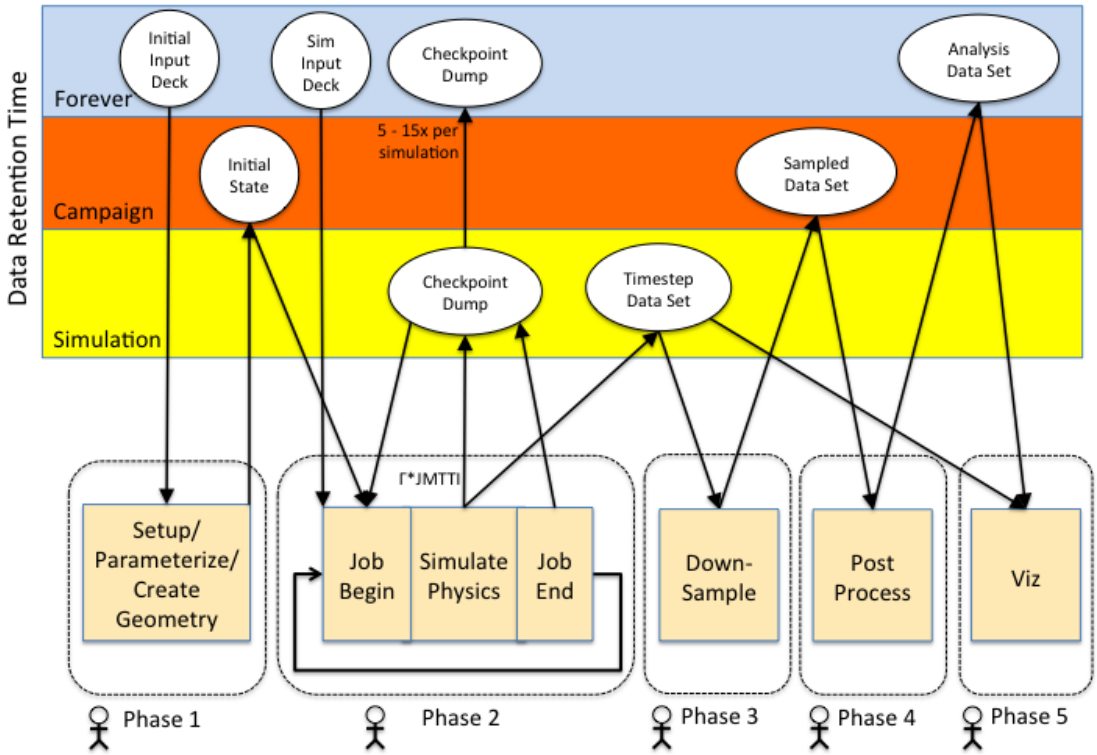

Figure 1: An example of an APEX simulation science workflow.

Figure 1 is an APEX workflow diagram that depicts the phases and data processing common to most simulation science workflows. In the initial phase we see that the scientist leverages a small input data set, typically curated over multiple campaigns, to construct a set of initial simulation conditions. For example, the resulting initial condition data sets are often a mesh representing a physical region of interest, where the physical regions of interest may range from the surface of the earth and atmosphere for climate modeling or massive regions of space to model galaxy collisions. Although phase 1 is only depicted once in the diagram, it may occur multiple times during a proposal team's campaign – though rarely more than 3 - 5 times per project. Further, the construction of the mesh is a parallel process and may use a moderate number of processors or it may use the entire machine. The typical constraint for initial conditions creation is acquiring the amount of memory to store the initial state and write it to file. Not surprisingly then, the generated input data set is usually a large portion of the memory of the processors allocated to phase 1, often 80% of the memory available to phase 1 jobs is written to storage.

Phase 2 is generally the most computationally intensive portion of the simulation science workflow. Given the initial state/mesh and an extremely small text-based input deck of settings and configurations, the physics calculations begin. The number of processors allocated to the job depend on the physics application's computational and memory requirements, and are typically static throughout the simulation. However, it is not uncommon for a project to use a coarse 2-dimensional mesh to create pilot data before running a much larger 3-dimensional mesh that spans a much larger portion of the machine. Restart dumps, or checkpoints, are created with a frequency high enough to ensure that when a job fails or finishes, a subsequent job can continue where the previous job progress ended. The ideal checkpoint interval, γ multiplied by the job mean time to interrupt (JMTTI) in the above diagram, has been estimated carefully by Daly [1], and most simulations create checkpoints with at least this frequency. Although a full system job might run for 24 hours and generate a checkpoint hourly, checkpoints are generally overwritten using an odd/even scheme. Thus when a phase 2 job successfully terminates 3 checkpoints should exist: an odd checkpoint, an even checkpoint, and an end-of-job checkpoint. In general, checkpoints are only used to start the next simulation job. However, checkpoints can often be analyzed for progress, and 5 - 15 checkpoints may be retained forever so that the portions of the simulation results can be re-calculated later for verification.

Phase 2 also results in the generation of analysis data sets. These data sets are generated at evenly spaced intervals in *simulated time*, but are typically not created uniformly throughout the life of the project or campaign. That is, the number of calculations required to construct analysis data sets varies over the duration of the simulation.

Phase 3 shows the common task of down-sampling the analysis data. By their very nature, all analysis data sets are critical to the scientists, however, once a data set is down-sampled, the data at the original resolution is typically too large to be retained (often in the range of 5 - 20% of the total allocated memory per data set). Down-sampling and analysis tasks in general are typically I/O intensive rather than compute intensive, and thus use the small numbers of compute nodes required to achieve the necessary memory footprint and adequate I/O bandwidth.

Phase 4 shows another common simulation science task, data post processing. Unlike phase 3, which is performed as the analysis data sets are generated, phase 4 requires all of the analysis data sets to exist prior to beginning phase 4. In this process the generated analysis data sets are formatted such that the regions of interest can be examined by visualization tools. This may involve concatenating portions of the data into separate data sets that favor visualization in isolation, or concatenating all of the data for visualization together. All post-processed data is vital to the scientist and will be retained beyond the lifetime of the campaign.

Phase 5 shows the single most important task within the simulation science workflow, interactive analysis. It is in this phase that the scientist uses tools such as Ensight, ParaView and VisIt to visualize and analyze the simulated and processed timestep data. It is only during this phase that scientific discovery can occur. Due to the importance of this step, interactive visualization typically occurs as soon as an analysis data set is generated, and then continues on through the life of the simulation and data processing phases.

1.3 Data Intensive Workflow Overview

This section describes the key features of the Uncertainty Quantification (UQ) and High Throughput Computing (HTC) workflows run on APEX machines. The commonality between UQ and HTC workflows is in the infrastructure that is needed to efficiently execute and collect results from an ensemble of runs. The software infrastructure used at Sandia is a UQ framework named Dakota (`https://dakota.sandia.gov/`) and at NERSC are various workflow management software (`https://www.nersc.gov/users/data-analytics/workflow-tools/`). The features provided by the software include

- **Efficient scheduling** of an ensemble of independent runs on a supercomputer. The work per run is often variable and so dynamic load balancing strategies are implemented to improve overall throughput. Queue wait time can also be significant and so multiple runs are often bundled together to improve overall throughput.
- **Collecting results** from each run in the ensemble.
- **Recovery from failures** when some or all of the individual runs die.
- **Detailed monitoring** of ensemble progress and saving provenance information.

The summarized workflow is shown in Figure 2. It can be applied to UQ and HTC.

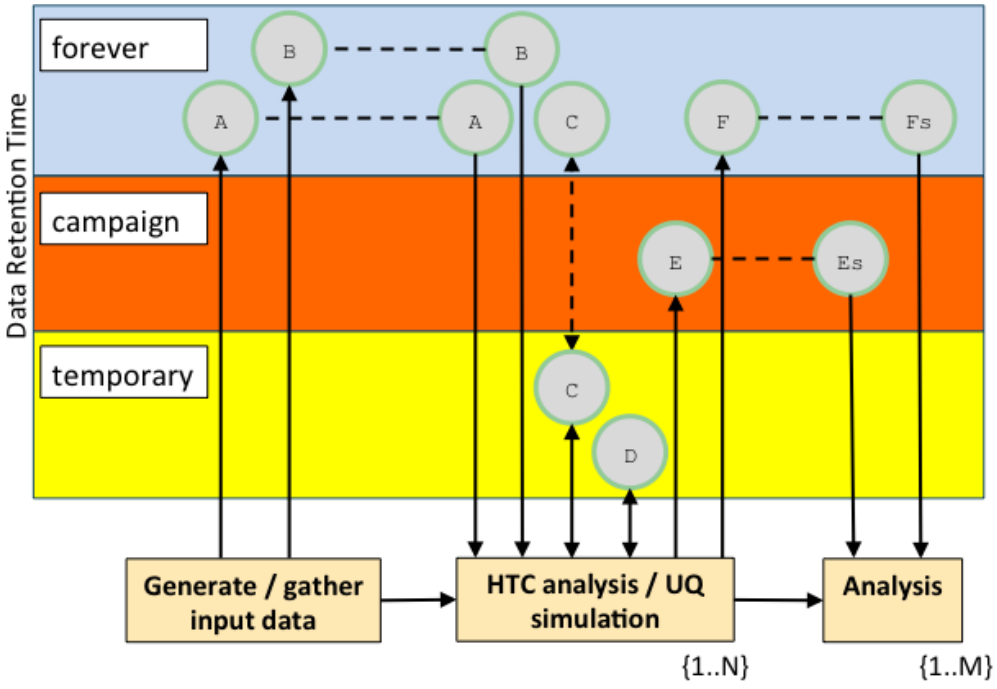

Figure 2: Example HTC and UQ workflow

A typical UQ workflow involves creating a mesh for the simulation (stage 1), running an ensemble of independent simulations (stage 2), and performing UQ analysis on the aggregated results to better understand the system being simulated (stage 3). The ensemble often

consists of 1-100 simulations which are each run with 20-500 MPI ranks and up to 10s of threads within each MPI rank.

A typical experimental HTC workflow also fits the same pattern. The Experimental or Observational Data (EOD) is sent to NERSC (stage 1), an ensemble of data analysis workflow pipelines are run on separate EOD data sets and the results (as well as location of raw and processed data) are aggregated into a database (stage 2). The N ensemble tasks may be executed together or staggered according to when EOD data sets arrive. The final database is made accessible through the web and is used for exploration and further analysis of data on time scales from minutes to years after the experiment or observation is performed (stage 3). The ensemble often consists of multiple applications chained together, where each application generally supports shared memory parallelism only.

The representative files are as follows

- **A**: A shared input file used by all tasks in the ensemble. This includes generated files such as an input mesh and gathered files such as reference lookup tables, e.g. detailed Equation of State (EOS) data or known genome sequences.
- **B**: An input file for each task in the ensemble. This includes parameter files and EOD data sets.
- **C**: An application checkpoint/restart file. There must be enough temporary storage to retain $3 \times N$ checkpoint files. Ensembles that run for a long time (more common in UQ) must restart from the checkpoint files. A subset of checkpoint files are saved forever.
- **D**: Temporary files used as input for dependent applications in the ensemble (more common in HTC) and to enable out-of-core algorithms.
- **E and F**: Files containing quantities of interest or processed data, e.g. images to make later analysis easier. The long-term value of the data dictates whether it should be stored for the lifetime of the campaign or forever.

The APEX machines should provide capabilities which help the UQ framework and workflow management software perform these tasks. It should be easy and fast to aggregate data, exchange progress data between the software and the ensemble, and reuse common data products in different runs of the ensemble, e.g. Equation of State (EOS) tables or bio-informatics reference databases (key-value stores).

1.4 Mapping the workflows to Cori/Trinity

In this section we describe possible ways to map the two workflows to the Cori/Trinity system. The Cori/Trinity system provides a storage system which contains SSD and HDD tiers. The SSDs are used to implement a Burst Buffer (BB) and the HDDs provide the storage for the Parallel File System (PFS). The product used to implement a BB on commodity SSDs is named Cray DataWarp. Details about DataWarp can be found in [2]. We briefly describe the functionality below.

The SSD storage in Cray DataWarp nodes is a scheduler resource which is dynamically assigned to compute jobs based on the amount requested in the user batch script. The storage can be assigned for the lifetime of the compute job: a job reservation, or independently of the compute job: a persistent reservation. The raw storage can be configured in a scratch mode, where files are explicitly moved between tiers using a command line or C API, or a cache mode, where file pages move transparently between tiers according to a user-defined caching policy. Finally, the user may choose a striped or private access mode where files are either striped across multiple DataWarp nodes or exist in a single DataWarp node. Most combinations of storage lifetime, configuration and access mode are supported.

1.4.1 Simulation science workflow

In our earlier description of the simulation science workflow we identified 3 general classes of data: input data sets, checkpoint data sets, and analysis data sets. Further, we identified the data retention periods of each storage system interaction for the associated data sets. In this section we examine how to efficiently create and interact with thse data sets on Cori/Trinity; however, we first must address the reliability of the burst buffer architecture. Fixed cost burst buffer architectures inherently make a tradeoff between performance and reliability. That is, for a fixed set of burst buffer storage media the highest performing burst buffer will have the least reliability, and a reliable burst buffer will perform more slowly than a less reliable burst buffer [3]. The same statement is true of parallel file systems (PFS); however, the parallel file system architectures used for Cori/Trinity are expected to experience data loss events very infrequently. In our description of how the workflows are mapped to the Cori/Trinity storage tiers, we will describe how our simulation science workflow mapping attempts to maximize scientific throughput and minimize rework in consideration of the unknown reliability of the burst buffer architecture.

Burst Buffer Interactions The burst buffer storage tier is intended to improve the performance of all large-scale storage system interactions. In general, all data sets large enough to require a significant storage interaction (e.g. an interaction longer than 10 - 30 seconds) should attempt to leverage the burst buffer. Thus Cori/Trinity's scheduler will include directives to pre-fetch initial data sets including initial input decks and mesh files, the checkpoint to use for simulation restart, and analysis data sets to be down-sampled, manipulated, or visualized.

The central role of the burst buffer is to improve the performance of scientific simulation checkpoint/restart functionality; and to that end we anticipate all generated checkpoints will be written into the burst buffer. As we described earlier, the odd/even/end-of-job checkpoint scheme ensures that the capacity demands of the checkpoint workload will be kept relatively low compared to the media endurance requirements. Further, the burst buffer will not migrate every checkpoint onto the PFS. Rather, we expect that one of the odd/even checkpoint dumps will be written from the burst buffer to the PFS, and that the end-of-job checkpoints will also be written from the burst buffer into the PFS.

The most complicated storage system interaction exists for analysis data sets. The burst buffer is not required to provide long-term reliability; however, analysis data is, by its nature, slated for long-term storage. Analysis data sets will be written into the burst buffer, and then

immediately written into the parallel file system (asynchronously to allow the application to continue its progress).

As we will discuss in the PFS tier interactions, a job has not truly finished successfully until the analysis data and any checkpoints to be retained are durable in the PFS. Nevertheless, the user will immediately schedule another job (usually the last line of a job batch script will submit the next job into the batch queue) with the hope that the next job will use the end-of-job checkpoint already extant. This allows the simulation to continue to make progress even though the "forever" data has not yet been written to its final locations.

Parallel File System Interactions The Cori/Trinity PFS is mounted by all compute nodes, thus applications will always have the option to bypass the burst buffer and use the file system directly. In fact we expect small-scale application pilot runs that work with small data sets are unlikely to use the burst buffer. Thus the PFS has the capability to synchronously read and write data in support of each of the five identified science phases. And if the burst buffer is temporarily unavailable due to outage, scientific progress can still continue by using the PFS, simply with less efficiency as the applications may spend significantly more time performing I/O than in the case when the burst buffer is functioning. In these cases, the user will perform the same cleanup manual actions that they do currently. The large majority of checkpoints will be deleted as soon as the analysis data from later timesteps is written into the PFS. A small number of checkpoints will be retained within the PFS until the campaign is completed, at which point those checkpoints will be written to long-term storage.

The more complicated PFS interactions (from a users point of view) occur when the burst buffer is used in conjunction with the PFS. At job completion, the next job is submitted to leverage the fact that the most recent checkpoint already exists within the burst buffer. However, the user cannot immediately perform manual cleanup operations for checkpoints generated by that job. Rather, until the analysis data sets are successfully migrated from the burst buffer into the PFS (which may happen considerably later than job completion), the preceding checkpoints cannot be deleted. On its surface, this is no different than the case where only the PFS existed as a storage tier for the simulation; however, the user's workflow must be adjusted as job completion is no longer a valid indicator that the analysis data and valuable checkpoints exist within the PFS. Instead, the user will have to manually verify that the data sets exist – which may lead to a larger PFS metadata read workload while users poll the file system to determine that cleanup may occur and rework is not necessary.

Long-term Storage Interactions Long-term storage interactions occur throughout the campaign period. As users generate initial states, identify valuable checkpoints, and generate analysis data sets, this data will be copied into an HPSS tape archive for long term storage. This interaction does not occur at prescribed intervals, rather users will backup data at an interval that makes sense based on the workflow progress and the size of the data sets. Large data sets will be backed up to long term storage immediately, whereas a series of small data sets may be accumulated into a usefully sized package before a backup copy is created. This off-system data movement is critical to achieving adequate protection for scientific data, but is not specific to Cori/Trinity.

1.4.2 Data intensive workflow

There are many opportunities in UQ and HTC workflows to reuse data already present in BB storage. However, as discussed earlier, reliability limits how long data products should remain in BB storage. For illustrative purposes below, we assume that data can be reliably stored in the BB.

We would like to use a persistent reservation of BB storage for the shared input files: **A**. These files are generally read by every single task in the ensemble and are also repeatedly read during a campaign. A job reservation of storage is not ideal because the data would need to be staged from the PFS to the BB many times.

We plan to use a job reservation of BB storage for the data products: **B, C and D**. The access mode, private or striped, depends on the file size and access frequency. In NERSC's HTC workflows, the size of the EOD data set: **B** is sometimes very small, e.g. < 100 MiB. The NERSC HTC workflows generally consist of multiple dependent applications which communicate through files. These files have no long-term value and are just needed to enable the workflow. We would use a scratch mode (and not a cache) for the temporary files because there is no need to move the files to the PFS. Checkpoint/restart libraries, e.g. HIO, will be used to manage the movement of checkpoint files: **C** between BB and PFS.

We would also like to use a persistent reservation of BB storage for the aggregated data products: **Es and Fs**. These output products will be accessed multiple times by different users long after the workflow pipeline has completed. It is helpful if the storage is managed as a cache for HTC experimental workflows because the data from the most recent experiment will be accessed more often than older data, but the older data will still be accessed. The accesses from the users include interactive analysis/visualization of raw data and simply looking at images of already analyzed data. The data to be saved forever will be copied to HPSS throughout the campaign. This data may be retrieved during the current or a future campaign.

2 Workflow Table

2.1 Table description

- **Workflow**: The name of the workflow. The LANL workflows are EAP, LAP, Silverton and VPIC; the NERSC workflows are ALS (which represents reconstruction of microtomography beamline data generated at ALS), GTS, HipMer, Materials (which includes Quantum Espresso and BerkeleyGW codes), MILC and Sky Survey; the SNL workflow is a typical UQ workflow enabled by Dakota.

- **Workload percentage**: The percentage of total facility compute time consumed by this workflow.

- **Representative workload percentage**: The percentage of total facility compute time consumed by this workflow and other similar workflows. For example, the Gyrokinetic Tokamak Simulation (GTS) workflow is assumed to represent the compute and storage needs of all nuclear fusion particle in cell (PIC) workflows run at NERSC.

- **Number of Cielo cores (Nc)**: The number of cores needed to run *today's* problem size on Cielo. A workflow often uses multiple concurrencies, however, we simplify things by just showing the number of cores used for the bulk of the computation.

- **Number of workflow pipelines per campaign (Np)**: The number of pipelines run during a single DOE allocation. In our analysis we assume that a single DOE allocation at NERSC is available for 12 months.

- **Anticipated increase in problem size by 2020**: The anticipated growth of datasets by 2020. For example, if a simulation is expected to use 8x more grid points by 2020 then this number will be 8. This number could be very low if the project plan is to incorporate additional physics or more detailed physics models into a simulation rather than increase mesh resolution.

- **Anticipated increase in number of workflow pipelines per campaign by 2020**: The anticipated growth in throughput by 2020. For example, if the number of simulations in a UQ ensemble is expected to grow by 2x then this number will be 2. This is an extremely important number to quantify the expected growth of experimental workflows run at NERSC.

- **Amount of data retained**: The capacity needed to store data used and produced by a single workflow pipeline. The capacity is given in units of percentage of system memory on Cielo. Use Equation 1 to convert the capacity from a percentage of system memory (Cm) to GiB (Cg) for a *single* problem size. Cielo has 2 GiB of DRAM memory per core (M).

$$Cg = \frac{Cm}{100} \times M \times Nc \tag{1}$$

  The duration that data must be stored is categorized into 3 groups: during a single pipeline, during a campaign, and forever. This is then further subdivided into different storage use cases: checkpoint/restart, analysis, read-only input, and out-of-core.

  - **During a pipeline (Tp)**: The capacity needed to store data which only needs to exist for the duration of a single workflow pipeline. Examples include temporary checkpoint files for resilience and restarting a simulation, temporary files for exchanging data between different workflow stages, and applications using out-of-core algorithms.
  - **During a campaign (Tc)**: The capacity needed to store data which must exist for the duration of a campaign. Examples include additional files for ad-hoc or unplanned data analysis.
  - **Forever (Tf)**: The capacity needed to store data which must be saved forever. This is the long-term storage requirement of the data used and produced in a single DOE allocation.

2.2 Table

The table is shown in Figure 3.

	LANL				NERSC						SNL
Workflow	**EAP**	**LAP**	**Silverton**	**VPIC**	**ALS**	**GTS**	**HipMer**	**Materials**	**MILC**	**Sky Survey**	**Dakota**
Workload percentage	**60**	**5**	**15**	**10**	**< 1**	**1**	**< 1**	**7**	**5**	**< 1**	**10**
Representative workload percentage	**60**	**5**	**15**	**10**	**3**	**6**	**7**	**19**	**11**	**3**	**10**
Number of Cielo cores	**65536**	**65536**	**143104**	**70000**	**100**	**16384**	**960**	**2400**	**100000**	**24**	**65536**
Number of workflow pipelines per campaign	**40 to 200**	**10 to 20**	**10 to 100**	**10 to 50**	**10760**	**100**	**?**	**100**	**1000**	**21000**	**50 to 200**
Anticipated increase in problem size by 2020	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**5.00**	**?**	**10 to 25**	**1.00**	**1.00**	**1.00**
Anticipated increase in workflow pipelines per campaign by 2020	**2.00**	**2.00**	**2.00**	**2.00**	**5.00**	**1.00**	**?**	**1.00**	**?**	**2.38**	**2.00**
Data retained (percentage of memory)	**240.00**	**2828.50**	**805.00**	**186.25**	**285.63**	**15.57**	**100.54**	**135.42**	**103.38**	**11.57**	**25.07**
During pipeline	**30.00**	**75.00**	**320.00**	**18.75**	**147.68**	**0.68**	**34.34**	**20.83**	**102.53**	**2.16**	**0.15**
Analysis					126.57		34.34	20.83		2.16	
Checkpoint	30.00	75.00	320.00	18.75		0.68					0.15
Input					21.10						0.00
Out-of-core									102.53		
During campaign	**60.00**		**80.00**	**37.50**	**21.10**				**0.62**		**0.00**
Analysis	60.00				21.10				0.62		
Checkpoint			80.00	37.50							
Input	0.00			0.00							0.00
Forever	**150.00**	**2753.50**	**405.00**	**130.00**	**116.84**	**14.89**	**66.20**	**114.58**	**0.23**	**9.41**	**24.93**
Analysis	50.00	2500.00	5.00	130.00	106.29	14.89	0.36	114.58	0.12	0.62	24.44
Checkpoint	100.00	250.00	400.00								0.49
Input		3.50			10.55		65.83	0.00	0.12	8.79	0.00

Figure 3: APEX workflow summary table. Note that this data may change as we continue our discussions with domain scientists

2.3 Example uses of table data

Q1. How much storage is needed to run a typical VPIC workflow pipeline?

The total data retained is 186.25% of memory at 70,000 cores. This is equal to 254.64 TiB (see Equation 1).

Q2. How much storage is used for productive I/O (i.e. not defensive I/O) in a typical VPIC workflow pipeline?

The total storage used for checkpoint/restart for resilience purposes is 18.75% of memory at 70,000 cores. This means that 229.00 TiB of storage is used for productive I/O.

Q3. How much long-term storage is needed to archive all data from the ALS workflow each campaign? (All NERSC data assumes that a campaign is a 1 year DOE allocation)

There are 10,760 workflow pipelines executed per year and the data saved forever corresponds to 116.84% of memory at 100 cores. This is equal to 233.68 GiB of data per workflow pipeline. This adds up to 2.40 PiB of data per campaign and consists of 0.22 PiB of beamline data and 2.18 PiB of analyzed data.

Q4. How is the Sky Survey workflow expected to change by 2020?

The growth in problem size is 1.00x and growth in throughput is 2.38x. The problem size is expected to remain constant because the resolution of the sky images will not change. This is because the resolution of Dark Energy Camera, which is used to collect the images, will not change. The throughput is increasing because the number of images is expected to grow from 1.26 million to 3 million.

Q5. How does the storage need of the HipMer workflow change depending on whether multiple workflow pipelines are scheduled simultaneously or consecutively?

We assume that we want to run ten workflow pipelines. If the workflow pipelines are run simultaneously then the storage requirement is 100.54% of memory at 960 cores all multiplied by 10 workflow pipelines. This is 18.85 TiB. If, on the other hand, the workflow pipelines are run consecutively then the storage requirement is 13.05 TiB (12.41 TiB for the long-term data products and 0.64 TiB as temporary scratch space for 1 pipeline at a time). The capacity difference could be significant if the user must schedule a fixed capacity of fast storage.

3 Closing

From the above described workflows the APEX team has mined many of the requirements used to draft our requirements for procuring compute resources. A careful reader of these workflows will likely find additional insights and generalizations that offer opportunities to improve the performance of how APEX campaigns are constructed from the constituent workflows.

4 Glossary of Terms

To better understand this document we provide the following glossary of terms used within this document.

Allocation A fixed quantity of computational time granted to a single project to use on the capacity-class computer.

Analysis Dump Data written from a parallel application for the express purpose of scientific analysis or visualization. The data is typically domain dependent and is typically analyzed in conjunction with other analysis dumps from the same parallel application (e.g. each dump may represent the system state at each second of the simulated system, with a total of 200 seconds simulated during a campaign).

Campaign A well-defined time interval during which a number of proposals will be allocated fixed quantities of computational time (i.e. allocations) on the capability-class computer.

Checkpoint Dump Data written from a parallel application that allows an interrupted application to resume from the progress made up to the time of the data creation. Simulations read the checkpoint dump at the beginning of a job in order to leverage prior progress toward simulation completion.

Data Retention Time Describes the time interval during which the data must be made durable. For example, an application that produces checkpoint dumps requires that those dumps must exist long enough for the application to progress far enough to create a subsequent checkpoint dump, or the application completes successfully. In the described workflows we describe three relevant retention times: immediate, meaning the data is temporary in nature and will be replaced with more up to date data as soon as possible; campaign, meaning the data is valuable throughout the campaign; and forever, indicating the data will outlive the life of the machine.

Gamma The optimal checkpointing time interval for a computational job. If JMTTI is 24 hours, Gamma=0.1 corresponds to an optimal compute interval between checkpoints of 2.4 hours.

Job A computational job runs on the capability-class machine and consumes the allocation of compute time.

Proposal A research proposal is a document that provides a detailed description of the manner in which the capability-class machine will be used to achieve scientific goals.

Simulation An execution of a parallel application that simulates physical phenomena. Simulations may take anywhere from hours to weeks or months to finish. A simulation will typically generate a series of restart dumps that allow the simulation to be composed into a series of jobs that execute on the capability machine.

Workflow A description of the dependencies and frequencies of a series of inter-related computational jobs. A proposal typically describes the constituent workflows required to support the proposed scientific inquiry.

Workload A description of how a capability-class machine executes workflows. The machine may co-schedule or interleave workflows to the degree possible within the workflow constraints.

References

[1] J.T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303 – 312, 2006.

[2] Cray. *DataWarp User Guide S-2558-5204*, September 2015. Available at `http://docs.cray.com/books/S-2558-5204/S-2558-5204.pdf`.

[3] John Bent, Brad Settlemyer, Nathan DeBardeleben, Sorin Faibish, Uday Gupta, Dennis Ting, and Percy Tzelnic. On the non-suitability of non-volatility. *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)*, 2015.